Методичка 9.3 - Знакомство с фреймворком radare2. Часть 2.

Обзор основных возможностей

В прошлом видео мы остановились на дизассемблировании функции main. Давайте посмотрим код функции main с помощью команды pdf (Print Disassemble Function).

Предварительно необходимо выполнить переместить указатель на начало функции main с помощью команды seek.

```
[0x08048370] s main
[0x080485bd] pdf
```

Давайте сейчас не будет разбирать каждую инструкцию функции main, мы это уже делали не один раз в этой курсе. Сейчас попробуем посмотреть на код функции в общем виде и составить псевдокод этой функции.

Спустя несколько инструкций мы видим вызов функции printf с сообщением “Usage:”. Видим, что перед вызовом функции есть условный переход.

```
    0x080485d0    833802        cmp dword [eax], 2
    0x080485d3      741d        je 0x80485f2
```

В качестве условного перехода используется инструкция JE. Значит, переход будет осуществлен, если адрес в регистре EAX будет указывать на число 2, иначе мы увидим строку “Usage: ...”. Выше видим, что значение в регистр EAX попадает из регистра ECX.

```
    mov eax, ecx
```

А еще выше мы видим, что в регистр ECX попадает адрес переменной arg_4 с типом int32.

```
    ; arg int32_t arg_4h @ esp+0x4
…
    lea ecx, [arg_4h]
```

В radare переменная arg_4 - это первый аргумент функции. А мы помним, что первый аргумент функции main - это argc. Значит сравнение осуществляется с переменной argc.

Напишем первые строки псевдокода.

```
if (argc == 2) {
   ...
} else {
        printf(“Usage: ...”)
```

}

Идем дальше. Далее мы видим прыжок в конец функции через инструкцию JMP. Значит на сообщении “Usage” функция завершает свою работу.

Добавляем еще одну строку в псевдокод.

```
if (argc == 2) {
    ...
} else {
        printf(“Usage: ...”);
        return;
    }
```

Далее мы видим вызов функции check_key и после вызова условный переход переход, в котором происходит проверка на значение 0 в регистре EAX. А мы помним, что после вызова функции в регистре EAX будет лежать значение, которое вернула функция.

```
call sym.check_key
…
test eax, eax
je 0x8048621
```

Снова в условном переходе используется инструкция JE. Значит переход будет осуществлен, если функция вернет нулевое значение.

Дополним псевдокод.

```
if (argc == 2) {
    ...
} else {
        printf(“Usage: ...”);
        return;
    }

    if (check_key() == 0) {
        ...
    } else {
        ...
    }
```

Идем дальше. Если переход состоится, то мы попадаем на строку “Congratulations! ...”, а если нет, то мы попадаем на строку “ERROR. ...”.

Дополним псевдокод.

```
if (argc == 2) {
    ...
} else {
        printf("Usage: ...");
        return;
    }

    if (check_key() == 0) {
        printf("Congratulations! ...");
    } else {
        printf("ERROR. ...")
    }
```

После того, как любая из этих строк будет напечатана, мы попадаем в конец функции main и функция завершает свою работу.

Дополним псевдокод.

```
if (argc == 2) {
    ...
} else {
        printf("Usage: ...");
        return;
    }

    if (check_key() == 0) {
        printf("Congratulations! ...");
        return;
    } else {
        printf("ERROR. ...");
        return;
    }
```

Итак. Мы написали псевдокод всей функции main. И теперь легко понять, что в коде две проверки. В самом начале проверяется количество аргументов, а затем вызывается функция ckeck_key и на основе значения, которое она вернет, будет напечатана строка.

В radare есть возможность посмотреть на код в псевдографическом режиме. Для этого необходимо выполнить команду - “V”.

В псевдографическои режиме есть несколько вариантов отображения кода. Менять их можно, если нажимать на кнопку “p”.

Перемещаться по коду можно либо стрелками, либо клавишами “k” и “l”. Если вы видите в коде адрес и хотите по нему перейти, то необходимо спустить до него и нажать кнопку “Enter”.

Обратите внимание, что для в функции main для каждого вызова функции в квадратных скобках есть число. Например, для функции printf - 1, для функции check_key - 2. Если мы нажмём на клавиатуре кнопку - 2, то сразу перейдем на код этой функции. Если мы хотим вернуться обратно в функцию main, то открываем режим команд через “:” (двоеточие) и пишем “s main”.

В этом режиме важно освоить переход на перекрестным ссылкам. По ним можно определить, есть ли в коде ссылки на определенный участок в коде.

Давайте перейдем внутрь функции puts - клавиша 3.

Нажмём на кнопку x.

Сверху видим, что было найдено 2 ссылки. Обе ссылки из функции main. Для перемещения между ними используем клавиши “j” и “k”.

Для того, чтобы перейти по ссылке нажимаем кнопку “Enter”. Как мы видим, мы оказались в коде функции main, где происходит вызов функции puts.

Различных команд существует очень много, с ними можно ознакомиться в документации к radare или через команду “?” (вопрос).

Выйти из псевдографического режима можно, если выполнить команду “q” (выход). И мы возвращаемся в режим консоли.

Есть еще интересный режим, который называется - режим графов. Для того, чтобы в него перейти необходимо выполнить команду “VV”.

И мы в консоли получаем визуализацию переходов и вызовов в функции. Перемещаться по графу можно с помощью клавиш - h/j/k/l. Для того, чтобы перейти в какую-нибудь функцию, вам необходимо посмотреть на буквы, которые указаны около функции в квадратных скобках и набрать их на клавиатуре. Например, для того, чтобы перейти к функции check_key необходимо набрать на клавиатуре - od. Для возвращения к функции main выполняем команду - “s main”.

Выйти из режима графов можно, если выполнить команду “q” (выход). И мы возвращаемся в режим консоли.

Как мы выяснили, ключ проверяет функция check_key, давайте выполним анализ этой функции.

```
[0x080485d0]> s sym.check_key
[0x08048585]> pdf
```

В самом начале мы видим, что в функции check_key только 1 входной аргумент и 3 локальные переменные.

```
; var char *s1 @ ebp-0x11
; var int32_t var_dh @ ebp-0xd
; var int32_t var_9h @ ebp-0x9
; arg char *s2 @ ebp+0x8
```

В начале функции мы видим пролог, затем происходит копирование строки “superkey” в локальную переменную s1.

```
mov dword [s1], 0x65707573  ; 'supe'
mov dword [var_dh], 0x79656b72 ; 'rkey'
mov byte [var_9h], 0
```

Затем указатель на строку кладется в стек и вызывается функция rot13. Rot13 - это шифр подстановки простой заменой для алфавита английского языка. Получается он преобразует строку “superkey”.

Далее происходит сравнение преобразованной строки со строкой, которая была передана функцию. Для сравнения строк используется функция strcmp.

```
push dword [s2]         ; const char *s2
lea eax, [s1]
push eax                ; const char *s1
call sym.imp.strcmp
```

Функция strcmp в случае успешного сравнения строк возвращает 0. Значит наша задача посчитать rot13 от строки “superkey”. В этом нам поможет утилита rahash2.

```
$ rahash2 -E rot -S s:13 -s ‘superkey’
‘fhcrexrl’
```

Проверяем ключ:

```
$ ./prog-7.2 fhcrexrl
```

Congratulations! License key is correct.

Отлично. Мы смогли узнать правильный ключ для программы prog-7.2.